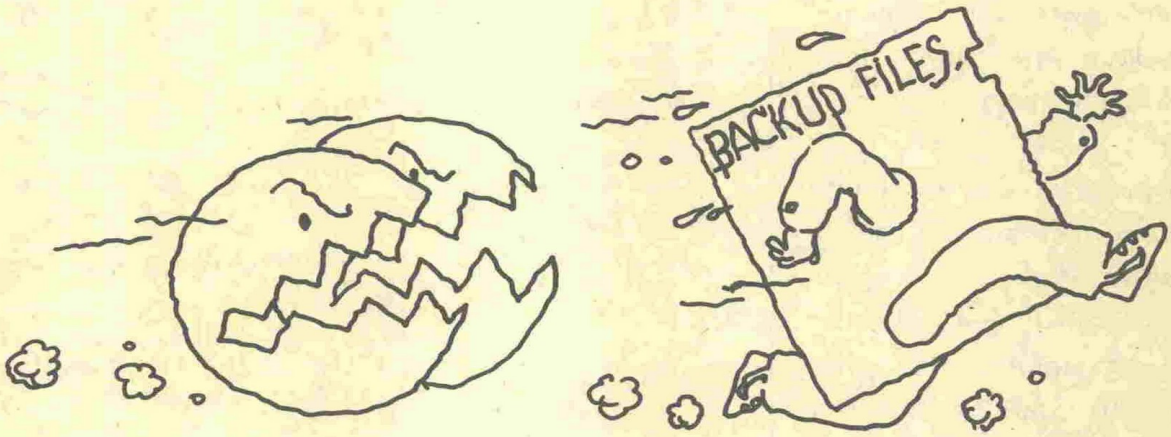
โปรแกรมช่วยลบ Backup Files



วีรวัจน์ บัวทอง

ที่ทำงานของผู้เขียนมี IBM AT COMPATIBLE และฮาร์ดดิสค์ 40 MB อยู่เครื่องหนึ่ง วันหนึ่งผู้เขียนลองลิสต์รายชื่อไฟล์ในฮาร์ดดิสค์ออกมาดู ก็พบว่ามี Backup File (ไฟล์ที่มีนามสกุล ".bak") อยู่ร่วม 40 ไฟล์ กระจัดกระจายอยู่ตามไดเรคทอรีต่าง ๆ กว่า 10 ไดเรคทอรีเกะกะไปหมด จึงตั้งใจจะลบไฟล์เหล่านั้นทิ้งให้หมด แต่พอลงมือลบจริงไปบางส่วน ก็เลิกล้มความตั้งใจในตอนแรกไป เพราะกว่าจะลบได้ต้องเสียเวลากระโดดไปตามไดเรคทอรีต่าง ๆ ทีละไดเรคทอรีซึ่งทำให้เสียเวลาไปไม่น้อย จากนั้นจึงเกิดความคิดที่จะหาโปรแกรมสำเร็จรูปที่เพียงแต่สั่ง

ครั้งเดียว ก็สามารถลบแบคอัพไฟล์เหล่านั้นออกได้หมด แต่หาเท่าไรก็ไม่พบ สุดท้ายจึงต้องพยายามเขียนขึ้นมาเอง

ก่อนที่จะเริ่มต้นเขียนโปรแกรม ผู้เขียนบังเอิญไปเห็นคำโฆษณาใน User's Guide ของภาษา Prolog ว่าสามารถใช้เขียนได้สั้นกว่าภาษา Pascal ถึง 10 เท่า ประกอบกับหัวใจสำคัญของโปรแกรมนี้ก็คือ "การค้นหาแบคอัพไฟล์ไปตามไดเรคทอรีต่าง ๆ" ซึ่งเป็นสิ่งที่ภาษาโปรล็อกชำนาญอยู่แล้ว ดังนั้น ผู้เขียนจึงตัดสินใจเขียนโปรแกรมนี้ด้วย Turbo Prolog version 2.0

โครงสร้างของโปรแกรม

โปรแกรม BAKDEL ที่เขียนขึ้นนี้ (โปรแกรมแสดงดังรูปที่ 1) ประกอบด้วย predicate หลัก (เทียบเท่ากับ procedure ในภาษาปาสคาล) 4 พรีดิเคต ดังนี้

1) **Predicate init** เป็นส่วนที่เซตค่าเริ่มต้นให้แก่โปรแกรม โดยจะเก็บไดเรคทอรีก่อนที่โปรแกรมจะถูกเรียกไว้ใน database ที่ชื่อ previous directory และเซตจำนวนไฟล์และจำนวนไบต์เริ่มต้น ให้เป็น 0 พร้อมทั้งพิมพ์ copyright message

2) **Predicate main** ประกอบด้วยพรีดิเคตย่อย ดังนี้

goto_root_directory : กระโดดไปยังไดเรคทอรีราก ของ default drive หรือ ไดเรคทอรีราก ของไดรฟ์ที่ระบุโดย command line paremeter

delebackup_if_exist : ตรวจสอบว่าในไดเรคทอรีขณะนั้น มีแบคอัพไฟล์หรือไม่ หากมีก็จะลบแบคอัพไฟล์ทั้ง

รูปที่ 1 โปรแกรม PBAKDEL.PRO ที่เขียนด้วยภาษาโปรล็อก

```
domains
   dirlist=string*

global predicates
   get_dirlist(string,integer,dirlist,integer,real) - (i,i,o,o,o) language c
   getcurdir(integer,string,integer)-(i,o,o) language c

database
   previous_directory(string)
   root_directory(string)
   total_files(integer)
   total_bytes(real)

predicates
   init
   main
   goto_root_directory
   valid_comline(string,integer)
   valid_comline
   delebackup_if_exist
   exist_backup(dirlist)
   dele_backup(dirlist)
   move_to_next_directory(dirlist)
   return_to_previous_directory
   write_result

goal
   init,
   main,
   write_result,
   return_to_previous_directory.

clauses
   init:-
      disk(Current),
      assert(previous_directory(Current)),
      assert(total_files(0)),
      assert(total_bytes(0)),
      write("BAKDEL-Backup File Deleting Utility Version 1.1, (C)Copyright 1989\n")

   main:-
      goto_root_directory,
      delebackup_if_exist,
      get_dirlist("*.*",16,List,_,_),!,
      move_to_next_directory(List).

   goto_root_directory:-            /* with command line paremeter     */
      valid_comline(Drive,Drive_number),
      getcurdir(Drive_number,Directory,Error),
      Error=0,
      concat(Drive,"\\",Root),
      assert(root_directory(Root)),
      concat(Root,Directory,Current),
      asserta(previous_directory(Current)),
      disk(Root).
   goto_root_directory:-            /* without command line parameter  */
      disk(Current),
      frontchar(Current,D,_),
      str_char(StrD,D),
      concat(StrD,":\\",Root),
      assert(root_directory(Root)),
      disk(Root).

   valid_comline(UpDrive,Drive_number):-
      comline(Drive),
      str_len(Drive,2),
      upper_lower(UpDrive,Drive),
      frontchar(UpDrive,Char,":"),
      Char>='A',Char<='Z',
      char_int(Char,Int_drive),
      Drive_number=Int_drive-64.
   valid_comline:-
      write("Invalid Command Line Parameter!\n"),
      exit(1).

   move_to_next_directory([H|T]):-
      disk(H),
      delebackup_if_exist,
      get_dirlist("*.*",16,List,_,_),!,
      move_to_next_directory(List),
      move_to_next_directory(T).
```

→

หมด พร้อมทั้งเก็บจำนวนและขนาดของไฟล์ที่ลบไว้ในดาต้าเบสชื่อ total_files และ total_bytes

get_dirlist : ค้นหาชื่อไฟล์ในไดเรคทอรีขณะนั้น แล้วส่งคืนให้ในรูปของ directory list พรีดิเคตนี้เขียนขึ้นด้วยภาษา C เนื่องจากในภาษาโปรลือกเองไม่มีคำสั่งที่เรียกใช้ได้โดยตรง

move_to_next_directory : กระโดดไปยังไดเรคทอรีถัดไป

3) **Predicate write result** จะพิมพ์ผลลัพธ์ว่าได้ลบแบคอัพไฟล์ ไปทั้งหมดกี่ไฟล์ ได้เนื้อที่ในดิสค์คืนมาทั้งหมดกี่ไบต์

ลักษณะของพรีดิเคตนี้ ถ้าจะเขียนด้วยภาษาอื่นจะเขียนได้ลักษณะดังรูปที่ 2

4) **Predicate return to previous directory** จะกระโดดกลับไปยังไดเรคทอรีเริ่มต้น ก่อนที่จะ run โปรแกรม Bakdel นี้

นอกจากนี้ยังมีพรีดิเคตย่อยในพรีดิเคตย่อย ทำหน้าที่ดังต่อไปนี้

Predicate valid comline เรียกโดย พรีดิเคต goto_root_directory เพื่อตรวจสอบว่า parameter ที่พิมพ์ตอนเรียกโปรแกรมครั้งแรก เป็นชื่อไดรฟ์ที่ถูกต้องหรือไม่ ถ้าถูกต้องก็ให้กระโดดไปลบไฟล์ในไดรฟ์ตามพารามิเตอร์นั้น ๆ ถ้าไม่ถูกต้องก็พิมพ์ error message จากนั้นจะ terminate โปรแกรม

การตรวจสอบว่าพารามิเตอร์ที่พิมพ์เข้ามาถูกต้องหรือไม่นั้น ใช้วิธีตรวจสอบว่าความยาวของพารามิเตอร์เท่ากับ 2 ตัวอักษร โดยที่ตัวอักษรตัวแรกเป็นตัวอักษร A ถึง Z และตัว

```
move_to_next_directory([]):-
   disk(Current),
   root_directory(Root),
   Current<>Root,!,
   disk("..").
move_to_next_directory([]).

return_to_previous_directory:-!,
   previous_directory(Dir1),
   disk(Dir1),
   retract(previous_directory(Dir1)),
   previous_directory(Dir2),
   disk(Dir2).
return_to_previous_directory.

delebackup_if_exist:-                /* backup file exists    */
   exist_backup(Backup_list),!,
   dele_backup(Backup_list).
delebackup_if_exist.                 /* backup file not exist */

exist_backup(Backup_list):-
   get_dirlist("*.bak",32,BackUp_list,Files,Sizes),
   not(Backup_list=[]),
   total_files(File),
   retract(total_files(File)),
   New_file=File+Files,
   assert(total_files(New_file)),
   total_bytes(Size),
   retract(total_bytes(Size)),
   New_size=Size+Sizes,
   assert(total_bytes(New_size)),!.

dele_backup([]).
dele_backup([H|T]):-!,
   deletefile(H),
   dele_backup(T).

write_result:-
   total_files(0),!,
   write("\nNo backup file found!").
write_result:-
   total_files(1),
   total_bytes(Sizes),!,
   write("\n","1 backup file has been deleted.\n"),
   writef("%f bytes recovered.\n",Sizes).
write_result:-
   total_files(Files),
   total_bytes(Sizes),!,
   write("\n",Files," backup files have been deleted.\n"),
   writef("%f bytes recovered.\n",Sizes).
```

```
CASE Total_files = 0
     WRITELN("No backup file found")
CASE Total_files = 1
     WRITELN("1 backup file has been deleted"),
     WRITELN(Total_bytes," bytes recovered")
OTHERWISE
     WRITELN(Total_files," backup files have been deleted"),
     WRITELN(Total_bytes," bytes recovered")
END CASE
```

รูปที่ 2 ลักษณะพรีดิเคตที่เขียนด้วยภาษาอื่น

อักษรตัวที่สุดท้ายเป็น colon(:) แต่ไม่ได้ตรวจว่าพารามิเตอร์นั้น เป็นชื่อไดรฟ์ที่มีอยู่ในระบบจริง ดังนั้น ถ้ามีการตั้งใจพิมพ์พารามิเตอร์ที่ผิด ๆ ก็จะเกิด run time error ขึ้นได้

Predicate exist backup เรียกโดยพรีดิเคต delebackup__if__exist ทำหน้าที่ตรวจสอบว่า ในไดเรคทอรีปัจจุบันมีแบคอัพไฟล์อยู่หรือไม่ โดยการเรียกใช้พรีดิเคต get__dirlist

ในกรณีที่มีแบคอัพอยู่ รายชื่อของแบคอัพไฟล์เหล่านั้นทั้งหมดจะถูกเก็บไว้ในลิสต์ที่ชื่อ Backup__list พร้อมกันนั้น จำนวนและขนาดของแบคอัพไฟล์ทั้งหมด จะถูกนำไปรวมกับจำนวนและขนาดของไฟล์ที่อยู่ในดาต้าเบส

ในกรณีที่ไดเรคทอรีปัจจุบันไม่มีแบคอัพไฟล์อยู่ พรีดิเคตนี้ก็จะให้ค่าเป็นเท็จ ดังนั้นโปรแกรมจะกระโดดถอยกลับ (backtracking) ไปยังพรีดิเคต delebackup__if__exist เงื่อนไขที่ 2 โดยไม่มีการกระโดดไปยังพรีดิเคต dele__backup เลย

Predicate getcurdir เป็นอีกพรีดิเคตหนึ่ง ที่เขียนโดยภาษา C เรียกโดยพรีดิเคต goto__root__directory พรีดิเคตนี้จะให้ค่าของไดเรคทอรีที่อยู่ปัจจุบันก่อนที่โปรแกรมจะกระโดดไปหาแบคอัพไฟล์ที่ไดเรคทอรีอื่น เพื่อว่าเมื่อได้ลบแบคอัพไฟล์ตามไดเรคทอรีต่าง ๆ ได้หมดแล้ว จะได้กลับไปยังไดเรคทอรีแรกเริ่มก่อนที่จะรันโปรแกรมนี้ได้

พรีดิเคตนี้ไม่มีในคำสั่งของภาษาโปรล็อกจึงต้องเขียนด้วยภาษา C แทน

Predicate dele backup เป็นพรีดิเคตที่เรียกใช้ต่อเนื่องมาจากพรีดิเคต

exist__backup โดยการนำเอาชื่อของแบคอัพไฟล์ในลิสต์ที่ได้จากพรีดิเคต exist__backup มาลบทีละไฟล์

การลบไฟล์ที่มีชื่ออยู่ในลิสต์สามารถทำได้โดยใช้เงื่อนไข 2 ข้อ ซึ่งเขียนโดยภาษาโปรล็อกได้ดังรูปที่ 3

เมื่อเริ่มรันพรีดิเคตนี้ โปรแกรมจะตรวจสอบโดยใช้เงื่อนไขข้อที่ 1 ก่อน

```
dele_backup([]).      /* ถ้าใน list ไม่มีสมาชิกอยู่เลย ให้เลิกลบได้ */

dele_backup([H|T]):- /* ถ้าใน list มีสมาชิกอยู่ สมาชิกตัวแรกคือ H */
   delefile(H),      /* สมาชิกที่เหลือคือ list ที่ชื่อ T ให้ลบ file */
   dele_backup(T).   /* H ออก จากนั้น เริ่มทำ predicate นี้ใหม่  */
                     /* ตั้งแต่ต้น โดยใช้ list เริ่มต้น คือ T      */
```

รูปที่ 3 การลบไฟล์ที่มีชื่ออยู่ในลิสต์

```
trace
domains
   strlist = string*
perdicates
   dele_backup(strlist)
   assume_deletefile(string)
goal
   dele_backup(["File1","File2","File3","File4"]).
cluases
   dele_backup([]).
   dele_backup(([H|T]):-
      assume_delefile(H),          /* Use assume_deletefile */
      dele_backup(T).              /* instead of delefile   */
   assume_delefile(File):-         /* for  demonstration    */
      nl,write(File),              /* purpose               */
      write(" has been deleted.").
```

รูปที่ 4 ตัวอย่างโปรแกรมที่ใช้ในการ trace

```
#include <stdio.h>
#include <dir.h>
#include <string.h>

#define listfno 1
#define nilfno  2

void *alloc_gstack(unsigned);
void *_malloc(int);

typedef struct ilist {
   unsigned char Functor;
   char *Name;
   struct ilist *Next;
} Dirlist;

void get_dirlist_0(char *mask,int attribute,Dirlist **Outlist,
                   int *total_file,double *total_size)
{
   struct ffblk ffblk;
   int done;
   int tfile=0;
   double tsize=0;
   done = _findfirst(mask,&ffblk,attribute);
   while(!done)
   {
      if ((ffblk.ff_name[0] != '.') && (ffblk.ff_attrib == attribute))
      {
         Dirlist *p=*Outlist=alloc_gstack(sizeof(Dirlist));
         p->Functor = listfno;
         p->Name = (char *) _malloc(13);
         _strcpy(p->Name,ffblk.ff_name);
         Outlist=&(*Outlist)->Next;
         tfile++;
         tsize +=ffblk.ff_fsize;
      }
      done = _findnext(&ffblk);
   }
   *total_file = tfile;
   *total_size = tsize;
   {
      Dirlist *p=*Outlist=alloc_gstack(sizeof(char));
      p->Functor=nilfno;
   }
}

char *path;

void getcurdir_0(int drive,char **outpath,int *error)
{
   *error=_getcurdir(drive,*path);
   *outpath=(char *)_malloc(MAXDIR);
   _strcpy(*outpath,*path);
}
```

รูปที่ 5 โปรแกรม CBAKDEL.C ที่เขียนด้วยภาษาซี

ในเงื่อนไขข้อที่ 2 โปรแกรมจะลบไฟล์แรกที่อยู่ในลิสต์ (ในที่นี้คือ H ซึ่งย่อมาจาก head) จากนั้นจะค่อย ๆ ลบไฟล์อื่น ๆ ถัดไปทีละไฟล์ (โดยการเรียกตัวเอง) จนกว่าลิสต์ที่เหลืออยู่จะกลายเป็นลิสต์ว่าง ทำให้เมื่อตรวจสอบว่า Backup__list เป็นลิสต์ว่างหรือไม่ ถ้าใช่ก็จะเลิกทำและไม่รันพรีดิเคตอื่นต่อไป แต่ถ้าไม่ใช่ลิสต์ว่าง เงื่อนไขข้อนี้จะเป็นเท็จ โปรแกรมจะกระโดดไปทำตามเงื่อนไขข้อที่ 2 ต่อไป

```
struct strlist {
   unsigned char Functor;
   char *Name;
   struct strlist *Next;
} Listname;
```

รูปที่ 6 string list ในรูปแบบของภาษาซี

ด้วยเงื่อนไขข้อที่ 1 จะได้ผลลัพธ์เป็นจริง พรีดิเคตนี้จึงจะหยุดทำงาน

สำหรับผู้ที่ไม่คุ้นเคยกับภาษาโปรล็อก ผู้เขียนอยากให้ลองดูลักษณะการทำงานของพรีดิเคตนี้ โดยการพิมพ์ตัวอย่างโปรแกรมสั้น ๆ ดังรูปที่ 4 แล้ว trace การทำงานของโปรแกรมดู จะทำให้เข้าใจได้โดยง่าย

get dirlist : พรีดิเคตที่เขียนด้วยภาษา C

เนื่องจากภาษาโปรล็อกไม่มีคำสั่งที่จะค้นหาชื่อไฟล์ได้โดยตรง ผู้เขียนในตอนแรกจึงตั้งใจจะใช้คำสั่ง bios interrupt 21H เพื่อค้นหารายชื่อของไดเรคทอรีและแบคอัพไฟล์ แต่มาภายหลังเห็นว่าในภาษา C มีคำสั่งซึ่งสามารถทำได้ จึงเกิดความคิดที่จะทดลองเขียนพรีดิเคตนี้ด้วยภาษา C แทนดังในรูปที่ 5 ในการเขียนพบว่าค่อนข้างจะลำบาก เพราะว่าในคู่มือของทั้งภาษาโปรล็อกและภาษา C แม้ว่าจะมีรายละเอียดการ interface ระหว่าง 2 ภาษานี้มากพอสมควร แต่กลับไม่อธิบายถึงการส่งผ่านลิสต์ที่มี member เป็น string เลย ดังนั้นผู้เขียนจึงต้องใช้วิธีทดลองผิดลองถูกอยู่นานพอสมควร กว่าจะสำเร็จได้

ก่อนที่จะทำความเข้าใจพรีดิเคตนี้ จะต้องเข้าใจวิธีการที่จะส่งผ่าน string list จากภาษา C ไปยังภาษาโปรล็อกก่อนดังนี้

string list ในภาษาโปรล็อกเมื่อเขียนเป็น structure ในภาษา C จะเป็นดังรูปที่ 6

โดยที่ Functor = 1 แสดงว่าสมาชิกในลิสต์ยังมีอีก

Functor = 2 แสดงว่าไม่มีสมาชิกในลิสต์อีกแล้ว

Name คือ pointer ที่ชี้ไปยังส่วนของ memory ที่เก็บตัวแปร string ไว้ และ Next คือ pointer ที่ชี้ไปยังสมาชิกของลิสต์ตัวถัดไป

ข้อสำคัญในการส่งผ่าน string list จากภาษา C ไปยังโปรล็อกก็คือเราจะต้องจองที่ของหน่วยความจำไว้สำหรับเก็บค่าของตัวแปรด้วย ซึ่งสามารถทำได้โดยใช้คำสั่ง malloc และ alloc_gstack

รูปแบบของการเรียกเข้าหาพรีดิเคต get_dirlist เป็นดังนี้

get_dirlist (Mask, Attribute, ReturnList, NumberOfFile, Totalbyte)

โดยที่ Mask = file หรือ directory mask

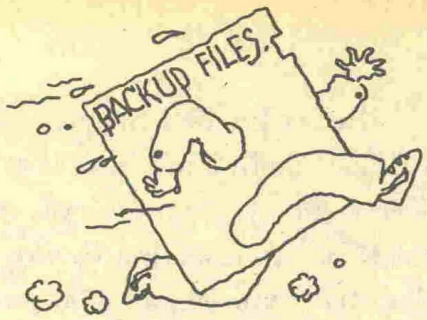
Attribute = 16 ในกรณีที่เป็น directory หรือ = 32 ในกรณีที่เป็น achieve file

ReturnList = ชื่อของลิสต์ที่เราต้องการให้มี สมาชิกที่มี Mask และ Attribute ที่ต้องการ

NumberOfFile = จำนวนไฟล์ทั้งหมดใน returnList

TotalByte = เนื้อที่ในดิสค์ทั้งหมด

เมื่อพรีดิเคตนี้ถูกเรียกว่าพารามิเตอร์ "Mask" และ "Attribute" จะถูกส่งผ่านให้แก่โปรแกรมย่อยที่เขียนด้วยภาษา C มีการเรียกใช้คำสั่ง findfirst และ findnext ของภาษา C เพื่อค้นหารายชื่อ directory หรือ file ที่มีชื่อตามค่าของ Mask เมื่อพบแล้ว ชื่อของไฟล์หรือไดเรคทอรีเหล่านั้นจะส่งคืนมาในรูปของ string list และในกรณีที่เป็นการหาชื่อของไฟล์ ขนาดและจำนวนของไฟล์ที่พบจะถูกส่งกลับมาพร้อมกันด้วย



การค้นหาชื่อของไฟล์หรือไดเรคทอรีนี้ จะค้นหาจนกว่าจะไม่มีไฟล์หรือไดเรคทอรีเหลืออยู่ ซึ่งโปรแกรมสามารถรู้ได้ โดยการตรวจสอบค่าของฟังก์ชัน findfirst หรือ findnext ของภาษา C ถ้าค่าของฟังก์ชันดังกล่าวเป็น 0 แสดงว่ายังมีไฟล์หรือไดเรคทอรีอื่นอยู่อีก แต่ถ้าเป็นค่าอื่นแสดงว่าหาชื่อได้ครบหมดแล้ว (ถ้าไม่เกิดความผิดพลาดอย่างอื่นเกิดขึ้น)

การคอมพายล์ให้เป็น EXE file

โปรแกรม Pbakdel และ Cbakdel จะต้องถูกคอมพายล์ให้เป็น OBJ file ก่อน โดย Cbakdel จะต้องคอมพายล์ด้วยโปรแกรม TCC.EXE ดังนี้

TCC-ml-O -c -u- -r- Cbakdel

ซึ่งจะได้เป็น module Cbakdel.obj ออกมา ส่วนโปรแกรม Pbakdel นั้น สามารถคอมพายล์ให้เป็น OBJ module ได้ภายใน integrate environment ของโปรล็อกเอง หลังจากที่คอมพายล์แล้ว ก็จะได้ module Pbakdel.obj และ Pbakdel.sym จากนั้นจะต้องนำ object module ทั้งสามมา link กันที่ command line ด้วยคำสั่งดังนี้

TLINK Init Pbakdel Cbakdel. sym,Bakdel,,Prolog + Emu + Mathl + Cl

โดยใช้ไฟล์ Init.Obj ที่มากับดิสค์ของ Turbo C เวอร์ชัน 2.0

การรันโปรแกรม

การรันโปรแกรม สามารถรันโดยมี command line parameter หรือไม่มีก็ได้ดังนี้

C > Bakdel ; ลบแบคอัพไฟล์ในไดรฟ์ C

หรือ

C > Bakdel d : ; ลบแบคอัพไฟล์ในไดรฟ์ D

โปรแกรม Bakdel นี้ใช้สำหรับการลบแบคอัพไฟล์ในดิสค์ที่มีไดเรคทอรีย่อยอยู่หลาย ๆ ชั้น โดยใช้การค้นหาแบคอัพไฟล์ในทุก ๆ ไดเรคทอรี ดังนั้น ถ้ามีไดเรคทอรีย่อย ๆ อยู่มาก โดยเฉพาะถ้าเป็นฟลอปปี้ดิสค์ก็จะเสียเวลานานพอสมควร (แต่ก็ยังเร็วกว่าและสะดวกกว่าการลบด้วยวิธีอื่นหลายเท่า) และเมื่อพบแบคอัพไฟล์แล้ว โปรแกรมก็จะลบไฟล์เหล่านั้นทันทีโดยไม่มีการเตือนก่อน ดังนั้นควรจะใช้ก็ต่อเมื่อมั่นใจว่า ไม่ต้องการแบคอัพไฟล์ทุกไฟล์แล้วจริง ๆ หากยังมีแบคอัพไฟล์ไหนที่ต้องการจะเก็บไว้ จะต้องเปลี่ยนนามสกุลของไฟล์ให้เป็นอย่างอื่นก่อน หรือเปลี่ยน attribute ของไฟล์ให้เป็นอย่างอื่นที่ไม่ใช่ archive

หวังว่าโปรแกรมนี้คงจะช่วยผู้อ่านทุ่นเวลาในการลบไฟล์ที่ไม่ต้องการ และพร้อมกันนั้นคงจะไม่ไปลบไฟล์ที่ต้องการใช้งานแถมไปด้วย